MERIDIAN
SOFTWARE SYSTEMS

Meridian Ada 4.1

Ada Graphics Utility Library

Using the Ada Graphics Utility Library

The Math Library source is located in the **agul** directory. To compile this source do the following:

1. Change to the **agul** directory.
2. Run **newlib**.
3. Run **buildit.bat** to compile and link the Graphics Library source and the demo sources called **agulmenu.ads**, **agulmenu.adb**, **aguldemo.ads**, **aguldemo.adb**, and **agul.ada**. These sources produce a test program called **agul.exe**. **Agul** demonstrates the Ada Graphics Utility Library.

You have now created an Ada program library that you can link to from your own application program libraries.

You can modify this source as you like for your own private use. You may not distribute or resale this source. You may distribute executable problems which have the Ada Graphics Utility Library linked into them without restriction.

TABLE OF CONTENTS

INTRODUCTION

SECTION 1

1.1 Purpose

The Ada Graphics Utility Library (AGUL) provides Ada programming language graphic support functions. These functions include screen and window operations and bitmap draw and fill figures. The bitmap figures include lines, circles, circle segments, arcs, ellipses, rectangles, object fills. The AGUL provides a structured and controlled method to successfully implement graphic functions within the Ada programming environment.

Some of the highlights of the Ada Graphics Utility Library include:

- Package **Graph_Initiate** - initialize the graphics system, puts the hardware into graphics mode, and determines the correct aspect ratio and pixel coordinate system to use. It also restores the original screen mode before graphics was initialized and frees the memory allocated on the heap for the graphics scan buffer.
- Package **Draw** - Graphic functions
- Package **Window** - Window Functions
- Package **Screen** - Screen Functions

1.2 Scope

It is assumed that the user of the Ada Graphics Utility Library is familiar with the Ada programming language and MSDOS. This document only explains the functionality of each function within the four packages previously mentioned. It is beyond of the scope of this document to provide information on the Ada programming language, low-level 80X86 interrupt functions, or the Meridian Software System AdaVantage compiler. Other document that may be of interest to the user are as follows:

- *ANSI/MIL-Std-1815A-1983, Reference Manual for the ADA Programming Language*, United States Department of Defense, Washington D.C.
- *ANSI/IEEE Std 983-1986, Ada As a Design Language*, The Institute of Electrical and Electronic Engineers, Inc., 1986
- Grady Booch, *Software Engineering with Ada*, Benjamin/Cummings Publishing Company, 1986
- J. G. P. Barnes, *Programming in Ada*, Addison-Wesley Publishing Company, 1984
- Grady Booch, *Software Components With Ada*, Benjamin/Cummings Publishing Company, 1987

- *AdaVantage DOS Environment Library, Meridian Software Systems, Inc*, Laguna Hills, CA, 1988
- Richard Wilton, *Programmer's Guide to PC & PS/2 Video Systems*, Microsoft Press, Bellevue, WA, 1987
- Peter Norton, *The Peter Norton Programmer's Guide to the IBM PC*, Microsoft Press, Bellevue, WA, 1985

1.3 Document Conventions

The description of each function within each package is organized alphabetically by package name and then function name. Each package is presented with its specification, a detailed description of its usage, several examples on how to use the function, and a figure to accompany each example.

PACKAGE COMMON_GRAPHIC_TYPES

SECTION 2

2.1 Specification

```
type Video_System is (
   Cga_Low_C4,     Cga_Low_Grey4,    Cga_High_C2,
   Evga_Low_C16,   Evga_Med_C16,     Evga_High_Bw,
   Evga_High_C16
   );

type Color is (
   Black,      Blue,           Green,        Cyan,
   Red,        Magenta,        Brown,        White,
   Grey,       Light_Blue,     Light_Green,  Light_Cyan,
   Light_Red,  Light_Magenta,  Yellow,       Bright_White
   );

subtype Display_Page is integer range 0 .. 1;

subtype Char_Horiz is integer range 0 .. 79;

subtype Char_Vert  is integer range 0 .. 53;

subtype Char_Pixel_Size is integer range 8 .. 16;

subtype Horizontal is integer range 0 .. 639;

subtype Vertical is integer range 0 .. 479;

subtype Index_Number is integer range 0 .. 8;

Aspect_Ratio            : Float;

Current_Page            : Display_Page;

Active_Page             : Display_Page;

Character_Pixel_Width   : Char_Pixel_Size;

Character_Pixel_Height  : Char_Pixel_Size;
```

2.2 Usage

The package Common_Graphic_Types contains type declaration for the Graph_Init, Draw, Window, and Screen packages.

2.2.1 Type Video_System

The enumeration elements of Video_System are described as follows:

- Cga_Low_C4 - 320x200 4 color (Video Mode 4)
- Cga_Low_Grey4 - 320x200 4 grey scale (Video Mode 5)
- Cga_High_C2 - 640x200 2 color (Video Mode 6)
- Evga_Low_C16 - 320x200 16 color (Video Mode 13)
- Evga_Med_C16 - 640x200 16 color (Video Mode 14)
- Evga_High_Bw - 640x350 monochrome (Video Mode 15)
- Evga_High_C16 - 640x350 16 color (Video Mode 16) 128KB Video Memory
- Evga_High_C16 - 640x350 4 color (Video Mode 16) 64KB Video Memory

2.2.2 Variables

Refer to Appendix A for the default settings for the following variables.

- Aspect_Ratio - set by Initiate_Graph routine. This value may be changed to adjust the angular drawings for your screen dimension.
- Current_Page - status variable only that is set by the Swap_Screen procedure. Initially set to 0 by the Initiate_Graph routine. This variable indicates which video page is visible. There are only two settings used 0 and 1 for video page 0 and page 1.
- Active_Page - status variable only that is set by the Select_Screen procedure. Initially set to 0 by the Initiate_Graph routine. This variable indicates which video page you can place text and drawings. There are only two settings used 0 and 1 for video page 0 and page 1.
- Character_Pixel_Width - initially set to eight (8) by the Initiate_Graph routine. This variable indicates the width of text characters in pixels.
- Character_Pixel_Height - this setting is determined by the video mode selected and is set initially by the Initiate_Graph routine. This variable indicates the height of text characters in pixels. It also indicates how many horizontal text lines there are for the current video mode.

PACKAGE DRAW

SECTION 3

3.1 Specification

```
with Common_Graphic_Types;

package Draw is

procedure Ellipse (
   XC  : Natural;
   YC  : Natural;
   A0  : Natural;
   B0  : Natural
   );

procedure Circle (
   XC  : Natural;
   YC  : Natural;
   R   : Natural
   );

procedure Circle_Segment (
   XC  : Natural;
   YC  : Natural;
   SA  : Natural;
   EA  : Natural;
   R   : Natural
   );

procedure Arc (
   XC  : Natural;
   YC  : Natural;
   SA  : Natural;
   EA  : Natural;
   R   : Natural
   );

procedure Rectangle (
   X1  : Natural;
   Y1  : Natural;
   X2  : Natural;
   Y2  : Natural
   );
```

```
procedure Line  (
   X1  : Natural;
   Y1  : Natural;
   X2  : Natural;
   Y2  : Natural
   );

procedure Object_Fill  (
   X  : Natural;
   Y  : Natural;
   Fil_C  :  Common_Graphic_Types.Color;
   Brd_C  :  Common_Graphic_Types.Color;
   );

procedure ForeGround_Color (
   Color  :  Common_Graphic_Types.Color
   );

procedure BackGround_Color (
   Color  :  Common_Graphic_Types.Color
   );

procedure Clear_Screen;

end Draw;
```

3.2 Usage

3.2.1 Procedure ARC

The arc portion of the circle segment is drawn by a chord-drawing method. Selected World Coordinates, window boundaries (if clipping is on), and screen limits are checked to ensure the ellipse is not drawn out of bounds. The arc is drawn with current foreground and background color.

```
Arc (xc, yc, sa, ea, r)
```

```
xc = horizontal center point in pixels
yc = vertical center point in pixels
sa = start angle in degrees (0 .. 360)
ea = end angle in degrees (0 .. 360)
r = radius, in pixels, from center point
```

3.2.1.1 Example

```
with Graph_Init, Draw;
use Graph_Init, Draw;

procedure Draw_Example is

  begin
    Graph_Init (evga_high_c16);

    Foreground_Color (Yellow);  --  change foreground color to yellow

    -- draw a yellow arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    Close_Graph;  --  close graphics and reset video mode to value
    --                 prior to starting the graphics mode.

  end Draw_Example;
```

3.2.2 Procedure Background_Color

This procedure will set the global variable Background_Color to the specified color.

This procedure is activated using the following calling convention:

Background_Color (color)

3.2.2.1 Example

```
with Graph_Init, Draw;
use Graph_Init, Draw;

procedure Draw_Example is

  begin
    Graph_Init (evga_high_c16);

    Background_Color (Light_Blue);  --  change background color to
    --                                  light blue

    Foreground_Color (Yellow);  --  change foreground color to yellow

    -- draw a yellow circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    Close_Graph;  --  close graphics and reset video mode to value
    --                prior to starting the graphics mode.

  end Draw_Example;
```

3.2.3 Procedure Circle

Draws a circle using the given center coordinates and radius in current foreground and background color.

This procedure is activated using the following calling convention:

```
Circle (xc, yc, r)
```

```
xc = horizontal center point in pixels
yc = vertical center point in pixels
r = radius, in pixels, from center point
```

3.2.3.1 Example

```
with Graph_Init, Draw;
Use Graph_Init, Draw;

procedure Draw_Example is

  begin
    Graph_Init (evga_high_c16);

    Foreground_Color (Yellow);  --  change foreground color to yellow

    -- draw a yellow circle with the center point at 400x75,
    -- and with a radius of 20 pixels from the two center points

    Circle (400, 75, 20);

    Close_Graph;  --  close graphics and reset video mode to value
    --                  prior to starting the graphics mode.

  end Draw_Example;
```

3.2.4 Procedure Circle_Segment

Draws an elliptical arc from the start angle to end angle using specified center and current color. The arc is then closed with two lines connecting the two arc endpoints to the center. Selected World Coordinates, window boundaries (if clipping is on), and screen limits are checked to ensure the ellipse is not drawn out of bounds.

This procedure is activated using the following calling convention:

```
Circle_Segment (xc, yc, sa, ea, r)


   xc = horizontal center point in pixels
   yc = vertical center point in pixels
   sa = start angle in degrees (0 .. 360)
   ea = end angle in degrees (0 .. 360)
   r = radius, in pixels, from center point
```

3.2.4.1 Example

```
with Graph_Init, Draw;
use Graph_Init, Draw;

procedure Draw_Example is

  begin
    Graph_Init (evga_high_c16);

    Foreground_Color (Cyan);  --  change foreground color to cyan

    -- draw a cyan circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    Close_Graph;  --  close graphics and reset video mode to value
    --                  prior to starting the graphics mode.

  end Draw_Example;
```

3.2.5 Procedure Clear_Screen

This procedure will clear the displayed video page (screen).

This procedure is activated using the following calling convention:

```
Clear_Screen
```

3.2.5.1 Example

```
with Graph_Init, Draw;
use Graph_Init, Draw;

procedure Draw_Example is

  begin
    Graph_Init (evga_high_c16);

    Foreground_Color (Red);  --  change foreground color to red

    -- draw a red circle with the center point at 400x75,
    -- and with a radius of 20 pixels from the two center points

    Circle (400, 75, 20);

    -- draw an red ellipse with the center point at 200x200,
    -- horizontal radius of 25 pixels,
    -- and with a vertical radius of 100 pixels

    Ellipse (200, 200, 25, 100);

    -- draw a red rectangle
    -- upper left horizontal pixel point at 100,
    -- upper left vertical pixel point at 50,
    -- lower right horizontal pixel point at 400,
    -- lower right vertical pixel point at 175

    Rectangle (100, 50, 400, 175);

    -- draw a red line
    -- beginning horizontal pixel point at 75,
    -- beginning vertical pixel point at 100,
    -- ending horizontal pixel point at 300,
    -- ending vertical pixel point at 75

    Line (75, 100, 300, 75);

    -- object fill - fill a circle
    -- horizontal pixel center point at 400,
    -- vertical pixel center point at 75,
    -- fill color - blue,
    -- border color - red

    Object_Fill (400, 75, blue, red);
```

3.2.7 ProcedureForeground_Color

This procedure will set the global variable Foreground_Color to the specified color.

This procedure is activated using the following calling convention:

```
Foreground_Color (color)
```

3.2.7.1 Example

```
with Graph_Init, Draw;
use Graph_Init, Draw;

procedure Draw_Example is

  begin
    Graph_Init (evga_high_c16);

    Foreground_Color (Yellow);  --  change foreground color to yellow

    -- draw a yellow arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    Foreground_Color (Cyan);  --  change foreground color to cyan

    -- draw a cyan circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    Close_Graph;  --  close graphics and reset video mode to value
    --                  prior to starting the graphics mode.

  end Draw_Example;
```

3.2.8 Procedure Line

A line is drawn as series of dots connecting the two end points. Coordinates of the two end points are specified and resolution of the displayed line is the selected resolution of the world coordinates and active window/screen.

Clip_Enable is checked to see clipping is desired for the current active window. If Clip_Enable is TRUE the procedure will determine whether the full length of a line can be drawn inside the active window. If part of the line can be drawn then the line coordinates are adjusted so that the line is inside the window.

This procedure is activated using the following calling convention:

```
Line (x1, y1, x2, y2)
```

```
x1 = horizontal starting point in pixels
y1 = vertical starting point in pixels
x2 = horizontal end point in pixels
y2 = vertical end point in pixels
```

3.2.8.1 Example

```
with Graph_Init, Draw;
use Graph_Init, Draw;

procedure Draw_Example is

  begin
    Graph_Init (evga_high_c16);

    Foreground_Color (Red);  --  change foreground color to red

    -- draw a red line
    -- beginning horizontal pixel point at 75,
    -- beginning vertical pixel point at 100,
    -- ending horizontal pixel point at 300,
    -- ending vertical pixel point at 75

    Line (75, 100, 300, 75);

    Close_Graph;  --  close graphics and reset video mode to value
    --                prior to starting the graphics mode.

  end Draw_Example;
```

PACKAGE GRAPH_INITIATE

SECTION 4

4.1 Specification

```
with Common_Graphic_Types;

package Graph_Init  is

procedure Initiate_Graph (
    Desired_Video : Common_Graphic_Types.Video_System
    );

procedure Close_Graph;

procedure Detect_Graphic_Card;

end Graph_Init;
```

4.2 Usage

4.2.1 Procedure Initiate_Graph

initialize the graphics system, puts the hardware into graphics mode, and determines the correct aspect ratio and pixel coordinate system to use. This procedure initializes the Ada Graphics Utility Library. It must be called before any other graphics procedure or function, but may only be called once within a program. Initiate_Graph selects the video page 0 as the active screen and erases it. All windows and worlds are initialized by calling Window_Reset and World_Reset.

This procedure is activated using the following calling convention:

```
Initiate_Graph (desired_video)


   desired_video - enumerated video_system type
```

4.2.1.1 Example

Refer to program example in Draw Package section.

4.2.2 Procedure Close_Graph

This procedure turns the graphics mode off and returns the system to text mode which was active before Initiate_Graph was called.

This procedure is called by using the following format:

```
Close_Graph;
```

4.2.2.1 Example

Refer to program example in Draw Package section.

4.2.3 Procedure Detect_Graph

This procedure dectects the video mode prior to initializing the graphic video system. It stores the detected video mode into a global variable to be used later by Close_Graph.

This procedure is called by using the following format:

```
Detect_Graph;
```

4.2.3.1 Example

Refer to program example in Draw Package section.

PACKAGE SCREEN

SECTION 5

5.1 Specification

```
with Common_Graphic_Types;

package Screen is

procedure Select_Screen (
    New_Page : Common_Graphic_Types.Display_Page
    );

procedure Copy_Screen;

procedure Swap_Screen;

procedure Save_Screen (
    Filename : String
    );

procedure Load_Screen (
    Filename : String
    );

end Screen;
```

5.2 Usage

5.2.1 Procedure Copy_Screen

This procedure will copy the currently displayed screen, either video page 0 or 1, to the nondisplayed screen, either video page 1 or 0.

NOTE

The existence of video page 1 is hardware dependent and will only be supported for EGA video systems. In the case of other graphics adapters, this procedure will have no effect.

This procedure is called by using the following format:

```
Copy_Screen;
```

5.2.1.1 Example

```
with Graph_Init, Draw, Screen;
use Graph_Init, Draw, Screen;

procedure Screen_Example is

  begin
    Graph_Init (cga_high_c2);

    Select_Screen (0);  --  set video page 0 as the active screen to
    --  place drawing or text on

    If Current_Page = 1 then
      Swap_Screen;  --  set the display page to video page 0
    end if;

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    --  copy video page 0 to video page 1

    Copy_Screen;
```

```
    -- clear video page 0

    Clear_Screen;

    -- display video page 1

    Swap_Screen;

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Screen_Example;
```

5.2.2 Procedure Load_Screen

Procedure Load_Screen reads a disk file that contains video data from a selected disk and loads it into the active screen.

Procedure Load_Screen uses the global variable Current_Page to determine which screen, video page 0 or page 1, will be loaded with the disk file data. It then uses the specified filename to determine the existence of the file on disk. If the file exists, Load_Screen loads it into the displayed video page. If the file does not exist, an error message is generated.

This procedure is called by using the following format:

```
Load_Screen (filename);
```

```
filename is the name of a disk file using the MS_DOS file naming con-
   vention.
```

5.2.2.1 Example

```
with Graph_Init, Draw, Screen;
use Graph_Init, Draw, Screen;

procedure Screen_Example is

  begin
    Graph_Init (cga_high_c2);

    Select_Screen (0);  --  set video page 0 as the active screen to
    --  place drawing or text on

    If Current_Page = 1 then
      Swap_Screen;  --  set the display page to video page 0
    end if;

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    --  store current display screen to a disk file

    Store_Screen(C:\StoreScn\NewScrn.Scn);

    -- clear video page 0
```

```
    Clear_Screen;

    --  load "NEWSCRN.SCN" disk file onto the current display screen

    Load_Screen(C:\StoreScn\NewScrn.Scn);

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Screen_Example;
```

5.2.3 Procedure Save_Screen

Procedure Save_Screen stores the active screen to a disk file. If the file does not exist, Save_Screen will create a file. If the file already exists, Save_Screen will overwrite the file.

Procedure Save_Screen uses the global variable Current_Page to determine which screen will be saved, video page 0 or page 1. It then uses the specified filename to check for the existence of the file on disk. If the file does not exist, it creates the file and saves the active screen to this file. If the file already exists, Save_Screen overwrites the old disk file with the active screen information.

This procedure is called by using the following format:

```
Save_Screen (filename);
```

```
filename is the name of a disk file using the MS-DOS file naming con-
   vention.
```

5.2.3.1 Example

```
with Graph_Init, Draw, Screen;
use Graph_Init, Draw, Screen;

procedure Screen_Example is

  begin
    Graph_Init (cga_high_c2);

    Select_Screen (0);  --  set video page 0 as the active screen to
    --  place drawing or text on

    If Current_Page = 1 then
      Swap_Screen;  --  set the display page to video page 0
    end if;

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    --  store current display screen to a disk file

    Store_Screen(C:\StoreScn\NewScrn.Scn);

    -- clear video page 0
```

```
    Clear_Screen;

    --  load "NEWSCRN.SCN" disk file onto the current display screen

    Load_Screen(C:\StoreScn\NewScrn.Scn);

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Screen_Example;
```

5.2.4 Procedure Select_Screen

This procedure determines which video page will be active either video page 0 or video page 1 (if present) for text and graphics placement.

NOTE

The existence of video page 1 is hardware dependent and will only be supported for EGA video systems. In the case of other graphics adapters, this procedure will have no effect.

This procedure is called using the following format:

```
Select_Screen (New_Page);


   New_Page = Video page number (0 or 1)
```

5.2.4.1 Example

```
with Graph_Init, Draw, Screen;
use Graph_Init, Draw, Screen;

procedure Screen_Example is

  begin
    Graph_Init (cga_high_c2);

    Select_Screen (1);  --  set video page 1 as the active screen to
    --  place drawing or text on

    If Current_Page = 1 then
      Swap_Screen;  --  set the display page to video page 0
    end if;

    -- draw the next two figures to video page 1
    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    -- clear video page 0
```

```
    Clear_Screen;

    -- display figures on video page 1

    Swap_Screen;

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Screen_Example;
```

5.2.5 Procedure Swap_Screen

Swap_Screen displays video page 1 if video page 0 is currently displayed or video page 0 if video page 1 is currently displayed. The active screen is not changed by Swap_Screen. This means that if you are drawing on one screen and call Swap_Screen while you are still drawing, the part of the drawing that has been completed is moved to the inactive screen, but subsequent drawing takes place on the active screen.

NOTE

The existence of video page 1 is hardware dependent and will only be supported for EGA video systems. In the case of other graphics adapters, this procedure will have no effect.

This procedure is called by using the following format:

```
Swap_Screen;
```

5.2.5.1 Example

```
with Graph_Init, Draw, Screen;
use Graph_Init, Draw, Screen;

procedure Screen_Example is

  begin
    Graph_Init (cga_high_c2);

    Select_Screen (0);  --  set video page 0 as the active screen to
    --  place drawing or text on

    If Current_Page = 1 then
      Swap_Screen;  --  set the display page to video page 0
    end if;

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    --  copy video page 0 to video page 1

    Copy_Screen;
```

```
    -- clear video page 0

    Clear_Screen;

    -- display video page 1

    Swap_Screen;

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Screen_Example;
```

5.2.6 Screen Package Program Example

```
with Graph_Init, Draw, Screen;
use Graph_Init, Draw, Screen;

procedure Screen_Example is

  begin
    Graph_Init (cga_high_c2);

    Select_Screen (0);  --  set video page 0 as the active screen to
    --  place drawing or text on

    If Current_Page = 1 then
      Swap_Screen;  --  set the display page to video page 0
    end if;

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    --  store current display screen to a disk file

    Store_Screen(C:\StoreScn\NewScrn.Scn);

    -- clear video page 0

    Clear_Screen;

    Select_Screen (1)  --  set video page 1 as the active screen to
    --  place drawing or text on

    Swap_Screen;  --  set the display page to video page 1

    -- draw an ellipse with the center point at 200x200,
    -- horizontal radius of 25 pixels,
    -- and with a vertical radius of 100 pixels

    Ellipse (200, 200, 25, 100);

    -- draw a rectangle
    -- upper left horizontal pixel point at 125,
    -- upper left vertical pixel point at 50,
    -- lower right horizontal pixel point at 300,
```

```
    -- lower right vertical pixel point at 200

    Rectangle (125, 50, 300, 200);

    --  copy video page 1 to video page 0

    Copy_Screen;

    -- clear video page 1

    Clear_Screen;

    -- display video page 0

    Swap_Screen;

    --  load "NEWSCRN.SCN" disk file and overwrite the current
    --  displayed screen

    Load_Screen(C:\StoreScn\NewScrn.Scn);

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Screen_Example;
```

PACKAGE WINDOW

SECTION 6

6.1 Specification

```
with Common_Graphic_Types;

package window is

procedure Define_Window (
   Window_Index : Common_Graphic_Types.Index_Number;
   WdwUpX  : Integer;
   WdwUpY  : Integer;
   WdwLwrX : Integer;
   WdwLwrY : Integer
   );

procedure Select_Window (
   Window_Index      : Common_Graphic_Types.Index_Number;
   Border_Color      : Common_Graphic_Types.Color;
   Window_Fore_Color : Common_Graphic_Types.Color;
   Window_Back_Color : Common_Graphic_Types.Color;
   Enable_Clip       : Boolean
   );

procedure Reset_Window;

procedure Save_Window (
   Window_Index : Common_Graphic_Types.Index_Number;
   Filename     : String
   );

procedure Load_Window (
   Window_Index : Common_Graphic_Types.Index_Number;
   Filename     : String;
   NewX         : Integer;
   NewY         : Integer
   );

procedure World_Coordinates (
   World_Index : Common_Graphic_Types.Index_Number;
   WldLwrX     : Integer;
   WldLwrY     : Integer
   );
```

```
procedure World_Select (
   World_Index : Common_Graphic_Types.Index_Number
   );

procedure World_Reset;

end WINDOW;
```

6.2 Usage

6.2.1 Procedure Define_Window

Define_Window is called to define a region of the screen as a window. The window definition consists of the upper left x and y coordinate and the lower right x and y coordinate. The window coordinates are sent to this procedure along with the index for the window. A maximum of eight windows can be defined at one time. This procedure will ensure that the window index is between one and eight inclusive. The coordinate data is checked for correct type and within the limits of the defined screen. A record is then created containing the window coordinates. If a previous definition has been stored at that index then it will be overwritten.

This procedure is called by using the following format:

```
Define_Window (Window_Index, WdwUpX, WdwUpY, WdwLwrX, WdwLwrY);
```

```
Window_Index = Index_number
WdwUpX = Upper Left column position
WdwUpY = Upper Left row position
WdwLwrX = Lower Right column position
WdwLwrY = Lower Right Y row position
```

6.2.1.1 Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
    -- clipping enabled

    Select_Window(2, blue, light_blue, bright_white, true);

    -- draw a circle_segment with the center point at 320x175,
```

```
   -- starting angle at 180 degrees,
   -- ending angle at 360 degrees,
   -- and with a radius of 100 pixels from the two center points

   Circle_Segment (320, 175, 180, 360, 100);

   -- draw a arc with the center point at 100x150,
   -- starting angle at 45 degrees,
   -- ending angle at 90 degrees,
   -- and with a radius of 50 pixels from the two center points

   Arc ( 100, 150, 45, 90, 50 );

   Close_Graph;  --  close graphics and reset video mode to value
   --  prior to starting the graphics mode.

end Window_Example;
```

6.2.2 Procedure Load_Window

Load_Window is called to load a window file from disk onto the displayed video page (screen). The window should be defined prior to calling the Load_Window procedure, if not an error message will be generated. This procedure retrieves the window coordinates from the window definition list. If the NewX and NewY coordinates are within the screen coordinates a temporary Lower_Right_X and Lower_Right_Y coordinates are calculated and checked against the screen coordinates. If the new coordinates are valid the window will be placed at the new coordinates. If the calculated coordinates are outside of the screen dimensions the window coordinates will be used. If the specified file exists the window will be loaded onto the displayed video page, otherwise an error message will generated.

The user can call Window_Load procedure using the following format:

```
Load_Window (Window_Index, Filename, NewX, NewY);
```

```
Window_Index = window index number
Filename = the name of a disk file using the MS-DOS file naming convention.
NewX = horizontal window offset
NewY = vertical window offset
```

6.2.2.1 Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
    -- clipping enabled

    Select_Window(2, blue, light_blue, bright_white, true);

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points
```

```
    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    --  store current window to a disk file

    Store_Window(C:\Window\NewWind.Wdw);

    -- clear video page 0

    Clear_Screen;

    -- turn windows off, clipping must always be false

    Select_Window(0, black, white, black, false);

    -- draw an ellipse with the center point at 200x200,
    -- horizontal radius of 25 pixels,
    -- and with a vertical radius of 100 pixels

    Ellipse (200, 200, 25, 100);

    -- draw a rectangle
    -- upper left horizontal pixel point at 125,
    -- upper left vertical pixel point at 50,
    -- lower right horizontal pixel point at 300,
    -- lower right vertical pixel point at 200

    Rectangle (125, 50, 300, 200);

    --  load "NEWWIND.WDW" disk file and reposition the window

    Load_Window (2, C:\StoreScn\NewWind.Wdw, 10, 10);

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Window_Example;
```

6.2.3 Procedure Reset_Window

The procedure will delete the list of Window_Records.

This procedure is called by using the following format:

```
Reset_Window
```

6.2.3.1 Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
    -- clipping enabled

    Select_Window(2, blue, light_blue, bright_white, true);

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    --  Clear the display screen

    Clear_Screen;
```

```
-- Reset the windows

Reset_Window;

-- draw the next figures on the display screen

-- draw an ellipse with the center point at 200x200,
-- horizontal radius of 25 pixels,
-- and with a vertical radius of 100 pixels

Ellipse (200, 200, 25, 100);

-- draw a rectangle
-- upper left horizontal pixel point at 125,
-- upper left vertical pixel point at 50,
-- lower right horizontal pixel point at 300,
-- lower right vertical pixel point at 200

Rectangle (125, 50, 300, 200);

Close_Graph;  --  close graphics and reset video mode to value
--  prior to starting the graphics mode.

end Window_Example;
```

6.2.4 Procedure Save_Window

Save_Window is called to save a specified window, on the displayed video page (screen), to a specified file on disk. The filename can include a specified disk drive. If the specified disk drive exists on the system the file will be written out to the disk. If the drive doesn't exist an error message is displayed. If the specified file exists the file is overwritten. This procedure will check that the specified window exists. If the window doesn't exist then an error message is displayed.

The Window_Save procedure can be called using the following format:

```
Window_Save (Window_Index, Filename);
```

```
Window_Index = window index number
Filename = the name of a disk file using the MS-DOS file naming convention.
```

6.2.4.1 Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
    -- clipping enabled

    Select_Window(2, blue, light_blue, bright_white, true);

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
```

```
   -- ending angle at 90 degrees,
   -- and with a radius of 50 pixels from the two center points

   Arc ( 100, 150, 45, 90, 50 );

   --  store current window to a disk file

   Store_Window(C:\Window\NewWind.Wdw);

   -- clear video page 0

   Clear_Screen;

   -- turn windows off, clipping must always be false

   Select_Window(0, black, white, black, false);

   -- draw an ellipse with the center point at 200x200,
   -- horizontal radius of 25 pixels,
   -- and with a vertical radius of 100 pixels

   Ellipse (200, 200, 25, 100);

   -- draw a rectangle
   -- upper left horizontal pixel point at 125,
   -- upper left vertical pixel point at 50,
   -- lower right horizontal pixel point at 300,
   -- lower right vertical pixel point at 200

   Rectangle (125, 50, 300, 200);

   --  load "NEWWIND.WDW" disk file and reposition the window

   Load_Window (2, C:\StoreScn\NewWind.Wdw, 5, 10);

   Close_Graph;  --  close graphics and reset video mode to value
   --  prior to starting the graphics mode.

end Window_Example;
```

6.2.5 Procedure Select_Window

Window_Select is called to select an indexed window from the list of window definitions. If a Window_Record exists on the stack the window will be drawn on the screen with the attributes passed to the procedure. The attributes include the border, foreground and background colors. The programmer may also specify clipping if desired. The selected window will then become the active window for any additional window commands.

If the indexed window doesn't exist an error message will be displayed on the terminal. If the indexed window exists the lines to be drawn will be drawn by calling the Rectangle procedure. The border lines will be drawn with the specified border color (if any) on the screen. The window background color will be fill within the window border. The Enable_Clip global boolean variable is set to TRUE to limit the drawings within the window from going outside the window boundaries.

This procedure is be called using the following format:

```
Select_Window (Window_Index, Border_Color, Window_Fore_Color, Window_Back_Color, Clip_Enable);
```

```
Window_Index = Window index number
Border_Color = color border around the window
Window_Fore_Color = window foreground color
Window_Back_Color = window background color
Clip_Enable = if enabled, the drawings within the window will be clipped
    at the window border, if not enabled, the drawings will extend beyond
    the window border.
```

6.2.5.1 Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
    -- clipping enabled
```

```
    Select_Window(2, blue, light_blue, bright_white, true);

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Window_Example;
```

6.2.6 ProcedureWorld_Coordinates

World_Coordinates defines a world coordinate system, by defining the vertices of the horizontal and vertical high values. The world coordinate system is not enabled until World_Select has been called. The maximum number of world coordinates that can be defined at one time is eight. The world coordinates are stored in a list of world coordinate records. The purpose of the world coordinate values is to allow for scaling of drawings within a window. The world coordinate system will only be used to define the window and not scaling factor for video page 0 or, if present, video page 1.

This procedure is called by using the following format:

```
World_Coordinates ( World_Index, WldLwrX, WldLwrY)

   World_Index = World index number
   WldLwrX = Lower right horizontal pixel coordinate
   WldLwrY = Lower right vertical pixel coordinate
```

6.2.6.1 Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- define world coordinate
    -- index number = 4
    -- lower right horizontal coordinate = 1280
    -- lower right vertical coordinate = 700

    World_Coordinate (4, 1280, 700);

    -- select a world
    -- index number = 4

    World_Select (4);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
```

```
   -- clipping enabled

   Select_Window(2, blue, light_blue, bright_white, true);

   -- draw a circle_segment with the center point at 320x175,
   -- starting angle at 180 degrees,
   -- ending angle at 360 degrees,
   -- and with a radius of 100 pixels from the two center points

   Circle_Segment (320, 175, 180, 360, 100);

   -- draw a arc with the center point at 100x150,
   -- starting angle at 45 degrees,
   -- ending angle at 90 degrees,
   -- and with a radius of 50 pixels from the two center points

   Arc ( 100, 150, 45, 90, 50 );

   Close_Graph;  --  close graphics and reset video mode to value
   --  prior to starting the graphics mode.

end Window_Example;
```

6.2.7 Procedure World_Reset

The procedure will delete the list of World_Coordinate.

This procedure is used by calling the following format:

```
World_Reset
```

6.2.7.1 Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- define world coordinate
    -- index number = 4
    -- lower right horizontal coordinate = 1280
    -- lower right vertical coordinate = 700

    World_Coordinate (4, 1280, 700);

    -- select a world
    -- index number = 4

    World_Select (4);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
    -- clipping enabled

    Select_Window(2, blue, light_blue, bright_white, true);

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);
```

```
    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    -- Reset the world coordinates

    Reset_World;

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Window_Example;
```

6.2.8 Procedure World_Select

World_Select selects a world coordinate system from memory to be the active World coordinate for any drawing commands that follow. This procedure may precede Window_Select to associate the world with the window.

The user can access this procedure in the following format:

```
World_Select (World_Index);
```

```
World_Index = World index number
```

6.2.8.1 Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- define world coordinate
    -- index number = 4
    -- lower right horizontal coordinate = 1280
    -- lower right vertical coordinate = 700

    World_Coordinate (4, 1280, 700);

    -- select a world
    -- index number = 4

    World_Select (4);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
    -- clipping enabled

    Select_Window(2, blue, light_blue, bright_white, true);

    -- draw a circle_segment with the center point at 320x175,
```

```
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Window_Example;
```

6.2.9 Window Package Program Example

```
with Graph_Init, Draw, Window;
use Graph_Init, Draw, Window;

procedure Window_Example is

  begin
    Graph_Init (evga_high_c16);

    -- define a window
    -- index number = 2
    -- upper left column at 10
    -- upper left row at 3
    -- lower right column at 30
    -- lower left row at 15

    Define_Window(2, 10, 3, 30, 15);

    -- define world coordinate
    -- index number = 4
    -- lower right horizontal coordinate = 1280
    -- lower right vertical coordinate = 700

    World_Coordinate (4, 1280, 700);

    -- select a world
    -- index number = 4

    World_Select (4);

    -- select a window
    -- index number = 2
    -- blue border color
    -- light blue background color
    -- bright white foreground color
    -- clipping enabled

    Select_Window(2, blue, light_blue, bright_white, true);

    -- draw a circle_segment with the center point at 320x175,
    -- starting angle at 180 degrees,
    -- ending angle at 360 degrees,
    -- and with a radius of 100 pixels from the two center points

    Circle_Segment (320, 175, 180, 360, 100);

    -- draw a arc with the center point at 100x150,
    -- starting angle at 45 degrees,
    -- ending angle at 90 degrees,
    -- and with a radius of 50 pixels from the two center points

    Arc ( 100, 150, 45, 90, 50 );
```

```
    --  store current window to a disk file

    Store_Window(C:\Window\NewWind.Wdw);

    -- clear video page 0

    Clear_Screen;

    -- turn windows off, clipping must always be false

    Select_Window(0, black, white, black, false);

    -- draw an ellipse with the center point at 200x200,
    -- horizontal radius of 25 pixels,
    -- and with a vertical radius of 100 pixels

    Ellipse (200, 200, 25, 100);

    -- draw a rectangle
    -- upper left horizontal pixel point at 125,
    -- upper left vertical pixel point at 50,
    -- lower right horizontal pixel point at 300,
    -- lower right vertical pixel point at 200

    Rectangle (125, 50, 300, 200);

    --  load "NEWWIND.WDW" disk file and reposition the window

    Load_Window (2, C:\StoreScn\NewWind.Wdw, 10, 5);

    -- Reset the world coordinates

    Reset_World;

    -- Reset the windows

    Reset_Window;

    Close_Graph;  --  close graphics and reset video mode to value
    --  prior to starting the graphics mode.

end Window_Example;
```

VIDEO GRAPHIC SYSTEM INFORMATION

APPENDIX A

CGA - 320 x 200 Four color and grey

- Aspect Ratio = 1.20
- Horizontal Screen Limits in Pixels = 0 - 319
- Vertical Screen Limits in Pixels = 0 - 199
- Video Page 0 Start Address = B8000
- Video Page 0 End Address = BBFFF
- Video Page 1 Start Address = not supported
- Video Page 1 End Address = not supported
- Character Pixel Width = 8
- Character Pixel Height = 8

CGA - 640 x 200 Two color

- Aspect Ratio = 2.41
- Horizontal Screen Limits in Pixels = 0 - 639
- Vertical Screen Limits in Pixels = 0 - 199
- Video Page 0 Start Address = B8000
- Video Page 0 End Address = BBFFF
- Video Page 1 Start Address = not supported
- Video Page 1 End Address = not supported
- Character Pixel Width = 8
- Character Pixel Height = 8

EGA - 320 x 200 16 color

- Aspect Ratio = 1.20
- Horizontal Screen Limits in Pixels = 0 - 319
- Vertical Screen Limits in Pixels = 0 - 199
- Video Page 0 Start Address = A0000
- Video Page 0 End Address = A7FFF
- Video Page 1 Start Address = A8000

- Video Page 1 End Address = AFFFF
- Character Pixel Width = 8
- Character Pixel Height = 8

EGA - 640 x 200 16 color

- Aspect Ratio = 2.40
- Horizontal Screen Limits in Pixels = 0 - 639
- Vertical Screen Limits in Pixels = 0 - 199
- If video memory equal to or less than 64 kilo bytes
  - Video Page 0 Start Address = A0000
  - Video Page 0 End Address = AFFFF
  - Video Page 1 Start Address = not supported
  - Video Page 1 End Address = not supported
- If video memory greater than 64 kilo bytes
  - Video Page 0 Start Address = A0000
  - Video Page 0 End Address = AFFFF
  - Video Page 1 Start Address = B0000
  - Video Page 1 End Address = BFFFF
- Character Pixel Width = 8
- Character Pixel Height = 8

EGA - 640 x 350 16 color (64KB) or 4 color (64KB)

- Aspect Ratio = 1.37
- Horizontal Screen Limits in Pixels = 0 - 639
- Vertical Screen Limits in Pixels = 0 - 349
- If video memory equal to or less than 64 kilo bytes
  - Video Page 0 Start Address = A0000
  - Video Page 0 End Address = AFFFF
  - Video Page 1 Start Address = not supported
  - Video Page 1 End Address = not supported
- If video memory greater than 64 kilo bytes

- Video Page 0 Start Address = A0000
- Video Page 0 End Address = AFFFF
- Video Page 1 Start Address = B0000
- Video Page 1 End Address = BFFFF

- Character Pixel Width = 8
- Character Pixel Height = 14

MCGA - 640 x 480 two color

- Aspect Ratio = 2.40
- Horizontal Screen Limits in Pixels = 0 - 639
- Vertical Screen Limits in Pixels = 0 - 479
- If video memory equal to or less than 64 kilo bytes
  - Video Page 0 Start Address = A0000
  - Video Page 0 End Address = AFFFF
  - Video Page 1 Start Address = not supported
  - Video Page 1 End Address = not supported
- If video memory greater than 64 kilo bytes
  - Video Page 0 Start Address = A0000
  - Video Page 0 End Address = AFFFF
  - Video Page 1 Start Address = B0000
  - Video Page 1 End Address = BFFFF
- Character Pixel Width = 8
- Character Pixel Height = 16

VGA - 640 x 480 16 color

- Aspect Ratio = 1.00
- Horizontal Screen Limits in Pixels = 0 - 639
- Vertical Screen Limits in Pixels = 0 - 479
- If video memory equal to or less than 64 kilo bytes
  - Video Page 0 Start Address = A0000
  - Video Page 0 End Address = AFFFF

- Video Page 1 Start Address = not supported
- Video Page 1 End Address = not supported

- If video memory greater than 64 kilo bytes
  - Video Page 0 Start Address = A0000
  - Video Page 0 End Address = AFFFF
  - Video Page 1 Start Address = B0000
  - Video Page 1 End Address = BFFFF
- Character Pixel Width = 8
- Character Pixel Height = 16

GLOSSARY

APPENDIX B

Absolute screen coordinate system: Coordinate system that uses the entire screen area to plot the pixel location of text or graphics; coordinate [0,0] is in the upper left corner of the screen.

Absolute variable: A variable declared to exist at a fixed location in memory rather than letting the compiler determine its location.

Active window: The window in which text placement or drawing is currently taking place.

Active screen: The video page (screen) in which text placement or drawing is currently taking place.

Actual parameter: A variable, expression, or constant that is substituted for a formal parameter in a procedure or function call.

Address: A specific location in memory.

Algorithm: A set of rules that defines the solution to a problem.

Allocate: To reserve memory space for a particular purpose, usually from the heap.

ANSI: The acronym for the American National Standards Institute.

Array: A sequential group of identical data elements that are arranged in a single data structure and are accessible by an index.

Argument: An alternative name for a parameter. See *actual parameter.*

ASCII character set: The American Standard Code for Information Interchange's standard set of numbers to represent the characters and control signals used by computers.

Aspect ratio: The horizontal-to-vertical ratio of a circle or ellipse. Used by the Ada Graphics Utility Library to proportion circles and circle segments.

Background: The screen surface and color on which text placement or drawing is taking place. See *foreground.*

Base point: Any of the points that constitute a graph or curve.

Bezier function: A function that uses an array of control points to construct a parametric, polynomial curve of predetermined shape.

Bit: A binary digit with a value or either 0 or 1.

Block: The associated declaration and statement parts of an Ada package specification or body.

Body: The instructions pertaining to a program or a subprogram (a procedure or function).

Bug: An error in a program. Syntax errors refer to incorrect use of the rules of the Ada language; logic errors refer to incorrect strategy in the program to accomplish the intended result.

Byte: A sequence of 8 bits.

Cartesian coordinate system: A method used to plot an object's location according to its horizontal-by-vertical position. This position is referenced to horizontal (X) and vertical (Y) axes.

Clipping: An Ada Graphics Utility Library function that keeps graphic images within window or screen boundaries by preventing any part of the drawing that falls outside the window or screen from being displayed.

Code: Instructions to a computer. Code is made up of algorithms.

Comment: An explanatory statement in the source code.

Control point: Any of the points used to plot a graph. Used by the Ada Graphics Utility Library to construct curves.

Coordinate system: A method used to plot an object's location according to its horizontal-by-vertical position. See *absolute screen coordinate system* and *world coordinate system.*

Declare: The act of explicitly defining the name and type of an identifier in a program.

Display screen: The visible screen displayed on the computer monitor.

Dynamic allocation: The allocation and deallocation of memory from the heap at runtime.

Dynamic variable: A variable on the heap.

Element: One of the items in an array.

Enumerated type: A user-defined scalar type that consists of an arbitrary list of identifiers.

File: A collection of data that can be stored on and retrieved from a disk.

File pointer: A pointer that tracks where the next object will be retrieved from within a file.

Flag: A variable that changes value to indicate that an event has taken place.

Foreground: The color used to display text and graphic images. See *background.*

Formal parameter: An identifier in a procedure or function declaration heading that represents the arguments that will be passed to the subprogram when it is called.

Global variable: A variable declared in a separate package than can be accessed from anywhere within the program.

Graphics Mode: The mode of computer operation in which graphics symbols and drawings are displayed. See *text mode.*

Heap: An area of memory reserved for the dynamic allocation of variables.

Inactive screen: The RAM or displayed screen that is not currently being used for drawing.

Initialize: The process of giving a known initial value to a variable or data structure.

Interrupt: The temporary halting of a program in order to process an event of higher priority.

Modeling: The method used to find the points (and the corresponding function) that will represent a predetermined line, curve, or solid shape. See *Bezier function.*

Origin: In any coordinate system, point [0,0], that is, the point where the coordinate axes intersect.

Pixels: An abbreviation for picture elements. The tiny dots that together make up a graphics or text screen display. Pixels are the basic units of measure used by coordinate systems to plot the location of screen objects.

Polar coordinate system: The method used to plot a circle segment in reference to its radius and the angle of its segment.

Polygon: A figure that encloses a collection of points, possibly (but not necessarily) connected by line segments.

Resolution: The quality and accuracy of detail of a displayed image. Resolution depends on the number of pixels within a given area of the screen; the more pixels there are, the higher the resolution.

Scaling: The ability of the Ada Graphics Utility Library to reduce or enlarge an image to fit in a given window according to the world coordinate system specified by the user.

Scope: The visibility of an identifier within a package.

Screen coordinate system: See *absolute screen coordinate system.*

Text mode: The computer mode in which only characters are manipulated and displayed. See *graphics mode.*

Vertex: The point where the sides of an angle intersect.

Window: An area of the screen specified by the user for displaying text or drawing.

Window definition coordinates: The two sets of *X* and *Y* coordinates that define the upper left and lower right corners of a window.

Window list: A linked list in which window definitions are stored.

World coordinate system: A user-defined coordinate system that is used to scale drawings within a given window. World *X* (horizontal) and *Y* (vertical) coordinates do not necessarily correspond to actual pixel locations, but can be assigned any values that suit the application. A world is enclosed by the *X* (horizontal) and *Y* (vertical) coordinates of the upper left and lower right corners of the drawing area.

Zero axes: The horizontal (*X*) and vertical (*Y*) axes used to plot the location of a screen object.